



下载APP



05 | 反爬虫的应对之策：通用且基本的检测规则是什么？

2022-01-28 DS Hunter

《反爬虫兵法演绎20讲》

课程介绍 >

**讲述：DS Hunter**

时长 17:50 大小 16.35M



你好，我是 DS Hunter。

上节课，我通过爬虫方的一整个抓取流程，给你讲了爬虫是如何低调地爬取站点，闷声发大财的。那么今天，我们就要看看反爬虫方要如何应对爬虫的抓取了。

我们都知道，当爬虫诞生的时候，反爬虫的需求就诞生了，而反爬虫这个职业，也就水到渠成，自然而然地出现了。那么我们要想进行反爬虫，要做的第一件事是什么呢？

没错，就是**识别爬虫**。爬虫如果无法被识别出来，那么剩下的所有架构设计以及扩展性都是在瞎扯。爬虫识别，可以说是整个反爬虫的核心。



这节课是反爬虫的第一课，因此我们只会聚焦一个问题：反爬虫通用且基本的检测规则，是什么？

通用检测规则

考虑到是通用且基础，我们先排除一些定制化的拦截检测，总结出如下几种检测方式：

1. TCP/IP 级别检测；
2. HTTP 级别检测；
3. 浏览器特征级别检测；
4. 业务相关特性检测。

这四条规则逐级递进，越往后，拦截越贴近应用，拦截效果越好；越靠前，对性能的影响越小，也就是性能越好。而较为特殊的拦截方式，我在第 9 和 10 讲也会给你提到一些。接下来，咱们就从 TCP/IP 级别检测开始了解吧。

TCP/IP 级别检测

TCP/IP 级别的检测，其实**主要是 IP 级别的检测，也就是俗称的“封 IP”**。它可以说是反爬虫的基础操作，人人都能想得到。

当然，它理论上也包含 TCP 检测，但是这个大部分公司是碰不到的，除非一些公司的爬虫有协议级别的错误。此外，App 爬虫大部分是直接走 TCP 协议，不过 App 爬虫我们在这门课程中是不讨论的。所以接下来，我们就看看 IP 级别的检测吧。

先说明一下，IP 级别的封锁，由于 IPv6 还在完善的路上，任重道远。所以，当前的用户主要是以 IPv4 为主。

在一个完整的网络请求中，IP 封锁可以选择在 SLB（服务器负载均衡，Server Load Balancing）层操作，也可以选择在业务层操作，各有好处。

	效率	功能
SLB层	✓	
业务层		✓

SLB层效率高，但功能少；业务层功能丰富，但效率低。



在 SLB 层操作，优点是非常彻底、高效。我们都知道，拦截越靠近业务层，拦截时机就越晚。而这个时候，服务器的压力已经产生了，性能会受到影响。那么，整体的拦截效果就会不明显。另一方面，SLB 层距离业务层有足够的距离，出于这种考虑，很多公司会在 SLB 做一层拦截。但是这一层的拦截也有坏处，那就是 SLB 功能有限，不能定制化得特别复杂，只能进行简单的规则定制。

而业务层则截然不同。**业务层的拦截，通常定制化极强。**因为业务层本身使用了复杂的编程语言来实现，可以实现任意逻辑，而不再是像 SLB 一样只能进行规则的配置。

最后我再给你说说**关于两个层封锁的区别和选择**。

举个例子：你要针对指定 IP 段进行封锁，但是出于业务需要，必须放掉一小部分（放掉一小部分的原因，我会在春节的时候给你展开讲讲）。这个在业务层，你只需要走一层黑名单，然后再走一层白名单进行召回就行了。但是，如果是在 SLB 层，它没有这么强的逻辑，可能就必须定制脚本了，难度直接飙升。这还只是一个简单需求，需求再复杂一点的话，SLB 可能根本无法完成。

那你可能就会问了：我到底应该选哪一个，才比较合适呢？嗯，实际上我们常说一句话：小孩子才做选择，成年人是全要的。是的，我这里的建议是一个都不能少，**两个方法都用上，相互配合封杀即可**。虽然可能会存在跨团队的情况，引发一些沟通成本，不过这是值得的。

了解了 SLB 层和业务层检测的不同之后，我就可以和你聊一聊怎么进行检测了。你可能认为，IP 封锁是一个很没技术含量的行为。这个想法其实并不正确。

IP 封锁一般来源于这样一个状况：你在网上搜索如何反爬虫，可以搜到的资料非常少，能搜到的基本也都是在教你怎么对 IP 频次进行检测并封锁。所以给人一种感觉：这个方法好 low 啊。但是，事实上这种办法只能叫基础，并不能叫做 low。基础和 low，在软件行业一直是两个不同的概念，不能划等号。

虽然说网上的办法，通常是对 IP 的频次进行检测然后封锁。可实际上，我们还能玩出更多的花样。

第一个是端口检测。我们都知道，网络上很多爬虫是通过代理爬取的。那么大部分的代理服务器为了让爬虫能连上去，都会开放一些端口给爬虫使用。一些低质的代理，通常开放的都是常用端口，比如 80 或者 8080。普通用户开放这类端口的概率极低，因此可以极大地增加嫌疑性。在 [03 讲](#) 一个演讲示范的故事中我们也提到过，没事扫扫端口，没准还有意外收获。

第二个是运营商检测，或者说 IP 段检测更好一些。我们都知道，有些爬虫会在公有云自己架设代理服务器，甚至有的节点干脆直接从公有云出来。对于这种请求我们没什么可客气的，整段都可以不留。

当然，这个方法的关键就在于检测 IP 段是否属于公有云。WHOIS 提供了足够多的信息，可喜的是，它还顺便把整个段都给了你。你可以选择用 range 直接封锁，也可以选择用前缀树来操作。两者相比，前缀树的可读性好一些，但是显得没有 range 专业，速度也比 range 慢一些，不过差别并不大。

之前我的实践中，SLB 层的封锁是使用 range 来做的，也就是子网，而业务方运营使用的是 IP 前缀树。因为 SLB 的操作人员通常是研发，所以面对子网没有任何障碍。而业务运营对 range，或者说子网，有一定的理解障碍。当然，这个要具体问题具体分析，效果是一样的。

第三个就是 SEO 的洗白了。反爬不是一个莽夫行为，一定不要神挡杀神，毕竟我们有一个善意的爬虫叫搜索引擎。我们还要给他们洗白，避免产生被拦截的情况。不过，搜索引擎这个东西呢，有的时候也会抽风，一旦抽风起来和一个 DDoS 也没什么区别.....所以洗白归洗白，我们也不能不管，这个度还是要把握一下的。

一个常见的做法是指定一个独立集群，如果 rDNS 之后认定为搜索引擎域名的，可以导出独立集群。这个集群可以不设置反爬，但是要设置资源上限，避免影响主业务。rDNS 相对比较靠谱，不用太担心冒充。

以上就是 TCP/IP 级别的几个检测，我给你介绍了在 SLB 层和业务层的检测方式以及各自的优缺点，这里，建议你搭配使用。同时，除了网上经常推荐的简单封 IP 操作，你也可以了解下那三个额外的补充操作，来帮助自己更有效更安全地对爬虫进行封杀。

下面，我们就看看更靠近应用层的 HTTP 检测。

HTTP 级别检测

HTTP 级别的检测主要集中在 HTTP 的 header。我们知道，**爬虫和普通用户唯一的不同之处就是，它的伪装一定有瑕疵。**HTTP 的 header 就是一个重灾区。

每一次浏览器的升级，都是爬虫的噩梦。因为浏览器升级经常会带来一些小的错误，尤其是 HTTP 的 header，甚至有的可能是拼写错误。举个最简单的例子，Chrome 的 Accept-Encoding，有个无用的属性叫 [SDCH](#)。这个属性是干嘛的，以及为什么说它无用，这里就不展开了，毕竟与反爬的关系不大。我们关键是通过 SDCH 来看，它为爬虫以及反爬虫带来了什么影响。

你可以在线上拉下所有 Chrome 用户的日志看一下，哪些用户的 Accept-Encoding 带 SDCH，哪些不带，以及哪些版本的 SDCH 拼写和别的版本不一样？带不带空格？都看完了，你就会发现，原来浏览器本身的问题也很多，像拼写、空格等等细节都有可能不同。

但是，这对反爬的人其实是好事，问题越多，爬虫学起来越累——浏览器本身就问题一大堆，还要 Cosplay 得一模一样，真是太痛苦了。

当然了，刚刚说的 SDCH 只是一个简单的例子，实际上所有 HTTP 的 header 都会多多少少有点问题，这些隐蔽的点都可以作为检测点。浏览器问题越多，模仿越有可能露出破绽。**而数据，无需手动收集，只要跑下线上用户日志即可。**

不过，在跑线上用户日志的时候要注意，**不要跑爬虫重灾区的服务地址的数据**，这样很可能把爬虫的错误当样本学习了。我们要尽可能用一些毫无意义的服务来提取样本。举个例子，价格页面，90% 的流量都可能是爬虫，这种流量有什么好学习的呢？这不是越学越坏

么？但是支付页面则不同，进去就付钱，正常哪个爬虫闲着没事来给你付钱？这个页面就是一个很好的样本点。

甚至，header 会有一些组合规律。这个可以直接用机器搞定，不用费心费力上人工。

除了 header 之外，有一些简单的 HTTP 格式也可以做一些检测。尤其是有一些低级的爬虫，经常犯一些 HTTP 请求格式写错的错误。这种你可以直接拦截掉，无需担心太多。当然，如果你发现一些畸形的 HTTP 请求触发了你服务器的 Bug，就要格外小心了。这可能不是低级爬虫，反倒是顶级爬虫，它在利用 SLB 的 Bug 来试图绕过反爬系统。

最后给你补充一个需要额外注意的地方：**各种 header 检测的正则表达式一定要考虑扩展性**。举个例子，近期 Chrome 版本号即将突破 100，很多老的反爬系统都被迫需要被修改，就是因为当时的扩展性没有留好。

浏览器特征级别检测

浏览器特征级别检测主要集中在 DOM。

我们先来看看大多数人的想法。很多人在反爬的时候，都期待有个唯一的 key 能让自己标记用户，于是就把希望放在了 Canvas 指纹上。不过，实际上 Canvas 指纹的冲突率还是挺高的，有的时候也会达不到我们期待的效果。

实际上，能够达到我们期待效果的 key 是存在的，你可以叫它 DOM 指纹。

单看反爬虫功能的话，我们不一定非要每个用户都有一个唯一的 key，这样做的话识别量反而更大了。如果所有的爬虫有一个 key，普通用户有另一个 key，那不是封杀起来更开心吗？我们只需要识别一次就好了。

至于具体的操作过程，其实就是从 Window 开始往下拉一棵树，循环引用跳掉，最终得到的 DOM 结构，就是一个指纹，这个指纹与浏览器相关。实际使用的时候，你可以使用部分 DOM，不要用全量 DOM，这样抓起来更灵活，底牌也可以一张一张慢慢打。

此外，**浏览器特征检测也可以利用一些浏览器 Bug**，但这种方式也有缺点，毕竟爬虫模拟得可能不够完善。那这种情况下，爬虫花费多大体力，你就对应要花费多少体力，未必是一个划算的买卖。

所以，在使用这个方法之前，我们就要仔细权衡了。毕竟，要进行浏览器特征的收集，这也是一个很大的体力活啊。爬虫那么多，我和它们一个个卷，我并不赚啊，怎么办？

别急，你要明白一件事情，你的目标从来都不是“弄清楚浏览器有哪些特征”，而是“**发现哪些请求的浏览器特征与其他人不一样**”。这个需求明确之后，你就会发现问题变得简单了很多。

想求两个数据集的 diff，这对于一个高级程序员来说，根本不是事。当数据量变大之后，无非就是如何优化降低复杂度、提升可靠性而已，最终将变成一个成本问题。所以，当一件事情从技术问题变成 ROI 问题之后，反倒简单了很多，你要做的只是打平收益和支出即可。

说到打平收益和支出，你可能要问了：那，我可以上机器学习吗？这样不就降低了人力成本，减少消耗，也就是减少支出了吗？

当然可以，没有任何问题，机器学习特别适合这种从混乱之中寻找规律的情况。但是，你一定要注意的是：机器学习本身解释性极差，如果发生了误伤，你很难给老板解释这一切为什么会发生，更不用说给客户解释了。

所以机器学习是可以上的，但是更多的是作为验证手段，或者让它替你发现一些特征。至于这些特征如何使用，一定要保证是人工控制，不能让机器替自己做决定。

业务相关特性检测

我在 [04 讲](#) 说过：反爬绝对不要做成中台。主要原因，就在这个检测上。

事实上，在数据下毒的过程中，使用的很多特性都是业务特性，中台方根本不知道如何给对手下毒。这么说吧，如果一个用户通常不会按照某个组合条件来查询你的商品，那么，你就可以在对应条件上对用户进行大幅降分，降低对用户的信任度。最后，如果用户再在某些条件上出现反常，就可以直接封杀了。

除了检测外，**数据处理上也是业务方更了解如何处理。**我之前就给你讲过 [“某公司爬取小说”](#) 的故事。他们一直都以为自己爬得很好，直到后来接到用户反馈，说章节错乱，小说都大结局了，你们这咋还差几章呢？检查了下发现对方一直给自己的是旧的章节。这类

下毒方式，就属于反爬虫方的业务特征下毒，而中台方因为并不了解具体业务，所以是很难想到的。

所以，在做这个检测的时候，你可以首先问自己一个问题：我们行业有什么奇葩之处？有没有什么我们看起来简简单单，但是外行看起来不可思议的业务特性？如果有，那这就是一个很好的监测点。

小结

今天我们一共介绍了四种不同的通用规则检测方式，分别是 TCP/IP 级别检测、HTTP 级别检测、浏览器特征级别检测以及业务相关特性检测。你也可以通过下面的图片回顾一下这些检测方式的具体使用方法：



这四种检测方式的具体的内容并不难理解，但是我们可以发现它们有一个共同的特点，那就是，**我们关心的往往不是用户到底是什么样子，而是用户与爬虫的差异点在哪里**。只有找到差异点，才能更好地区分、检测，最后进行拦截。

至于这个差异是如何产生的，我们未必要穷追到底，只需能够利用即可。这些基础的封锁都做掉之后，爬虫再也不能低调的偷偷爬数据了，不得不和你正面开打。

目前为止，爬虫和反爬的斗智斗勇，还只能认为是打架斗殴。一旦规模上来了，我们才能称之为战争。

下一讲，我们将看到，分布式的爬虫是怎么和反爬虫方开战的。这样，爬虫和反爬虫的战争，就正式打响了。

思考题

好了，又到了愉快的思考题时间，还是三选一的老规矩，记得保密脱敏哦：

1. 我们知道任何线上操作，风险都与收益并存。你认为封 IP 这种最基础的操作，风险和收益分别是大还是小？具体风险和收益分别是什么？
2. 我们可以通过浏览器特征检测爬虫，那么如果爬虫固定一个版本的浏览器，我们其余的检测岂不是徒劳的？如何解决这个问题呢？
3. 检测的东西如何对爬虫方保密呢？毕竟他们一旦看到了，就知道如何绕过。

期待你在评论区的分享，我会及时回复你。反爬无定式，我们一起探索。

“

基石总归要修筑，息壤依然是主流。



反爬虫兵法演绎 20 讲

DS Hunter / 反爬虫专家，前某公司研发总监

”



识别二维码
免费试读



分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

 生成海报并分享

 赞 1  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 04 | 爬虫基础理论概述：如何低调地拿到自己想要的数据？

下一篇 06 | 爬虫应对之策：分布式的意义在哪里？

精选留言 (3)

写留言

13 宫十三 
2022-02-04

- 1 相关经验少，就不班门弄斧了
- 2 固定版本有两个对抗思路，一是高频，如果某个版本的流量趋势激增，能够说明流量有水分了；二加入DOM检测，多维度验证
- 3 核心代码保护，也叫代码加密

作者回复：赞，基本思路没问题了。



 **webmin** 
2022-01-30

1. 静态封IP的话滞后太多且效果不好，动态封IP的话算法好不容易误伤，特别是某位BOSS被误伤的时候，现在NAT的情况比较多，需要黑白名单一起使用；
2. 浏览器版本只是一个特征点，或者说多重过滤中一重吧；
3. 社会工程学，分别让不同的人知道不同的种特定防爬策略，通过哪一种策略失效来找到泄密口子或方向，验证几次大约就能定位出问题出在哪里；

展开 ∨

作者回复：哈哈，某位boss，感觉很有故事啊。尤其是boss和白名单放一起说。。。。



 **伟**
2022-01-30

dom指纹没太听懂，如何生成及使用能展开给我们详细讲一下吗？是每个页面都有不一样的dom指纹么？然后所有的用户在同一个页面上的dom指纹其实是一样的？怎样防止dom指纹被伪造，反爬怎么进行拦截，是没有dom指纹的就直接拦截吗？

作者回复: 首先指纹这个词就是用来标识唯一的，不是用来防伪造的。所以dom指纹就是dom特征的唯一标识。防伪造靠加密。所以，取多个dom特征，甚至全部特征，就可以了。

